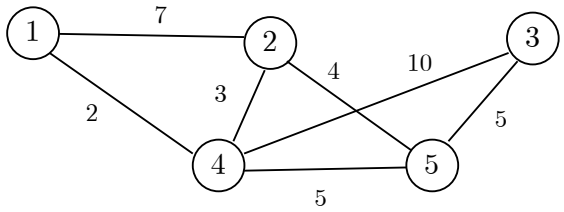
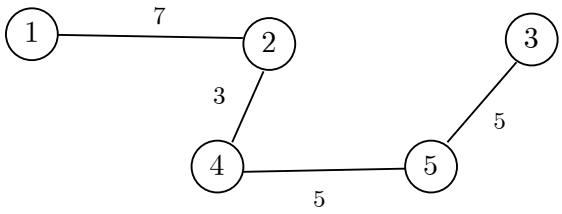
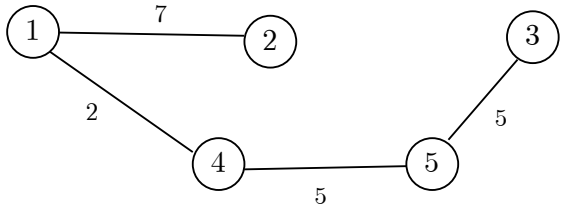
Minimum spanning tree

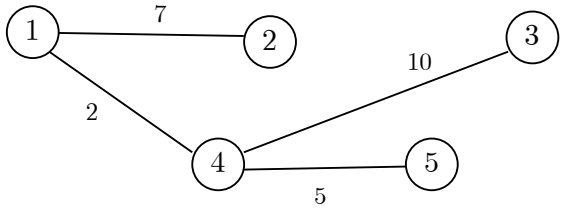
Spanning tree คือ ต้นไม้ที่ประกอบด้วยโหนด (Node) ทุกโหนดของกราฟ โดยแต่ละคู่ของโหนดใด ๆ จะมีเส้นเชื่อม (Edge) เพียงเส้นเดียว ดังนั้น Spanning Tree จะไม่มี Loop หรือ Cycle



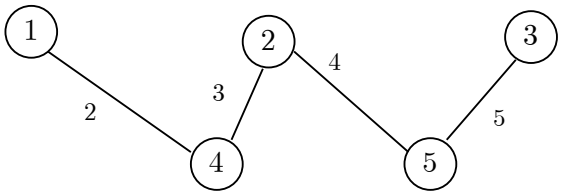
จากตัวอย่างกราฟข้างบนสามารถสร้าง Spanning tree ได้หลายรูปแบบ เช่น







Minimum spanning tree คือ Spanning tree ที่มีผลรวมของค่าน้ำหนักของเส้นเชื่อมที่มีค่าน้อยที่สุด ดังนั้น จากกราฟข้างบนจะได้ Minimum spanning tree ดังต่อไปนี้ โดยมีน้ำหนักรวมของเส้นเชื่อมเท่ากับ 14



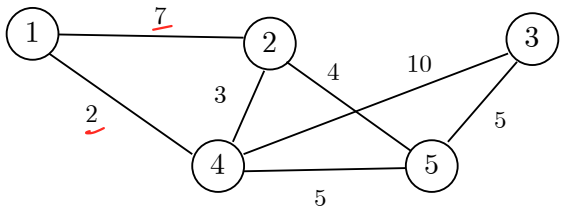
Prim's Algorithm

แนวคิดของ Prim's algorithm คือ เริ่มจากกำหนดโหนดเริ่มต้นของ Spanning tree แล้วพิจารณาว่ามีเส้นเชื่อมของโหนดดังกล่าวไปยังโหนดใดบ้าง (โดยโหนดที่เชื่อมไปจะต้องไม่อยู่ใน Spanning tree) และทำการเลือกเส้นเชื่อมที่มีน้ำหนักน้อยสุดเพื่อนำโหนดที่เชื่อมไปมาสร้างต่อใน Spanning tree หลังจากนั้นก็ทำการพิจารณาเหมือนเดิมแต่ให้เลือกเส้นเชื่อมที่น้อยที่สุดจากทุกโหนดที่อยู่ใน Span tree ทำอย่างนี้ไปเรื่อยๆ จนกระทั่ง Spanning tree มีโหนดครบทุกโหนด

The steps for implementing Prim's algorithm are as follows:

1. Initialize the minimum spanning tree with a vertex chosen
2. Find all the edges that connect the tree to new vertices, then select the weight minimum and add it to the tree
3. Repeating step 2 until we get V nodes in spanning tree

ตัวอย่าง การสร้าง Minimum spanning tree ของกราฟ G ด้วย Prim's algorithm

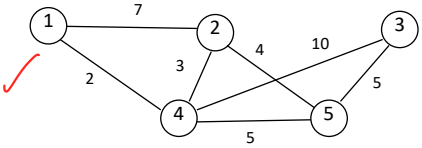


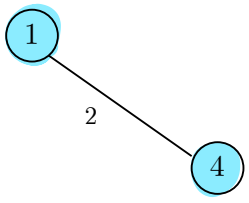
เริ่มต้นด้วยการใช้โหนด 1 ในการสร้าง Spanning tree

พิจารณาโหนดใน Spanning tree ซึ่งตอนนี้ใน Spanning tree มีเพียงโหนดเดียวคือ โหนด 1 ดังนั้นในการขยาย Spanning tree ให้พิจารณาโหนดที่เชื่อมต่อกับโหนด 1 ซึ่งคือโหนด 2 และ 4 แล้วเลือกโหนดที่เชื่อมต่อด้วยค่าน้ำหนักที่น้อยที่สุด

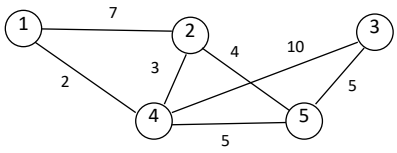
เส้นเชื่อมระหว่างโหนด 1 กับโหนด 2 มีค่าน้ำหนักเท่ากับ 7

เส้นเชื่อมระหว่างโหนด 1 กับโหนด 4 มีค่าน้ำหนักเท่ากับ 2

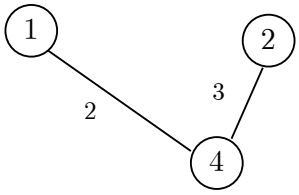




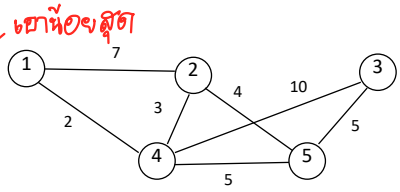
พิจารณาโหนดใน Spanning tree ซึ่งตอนนี้ใน Spanning tree มีโหนด 1 และ 4 ดังนั้นในการขยาย Spanning tree ให้พิจารณาโหนดที่เชื่อมต่อกับโหนด 1 และ 4 ซึ่งคือ โหนด 2 3 และ 5 (เฉพาะโหนดที่ไม่ได้อยู่ใน Spanning tree) แล้วเลือกโหนดที่เชื่อมต่อด้วยค่าน้ำหนักที่น้อยที่สุด



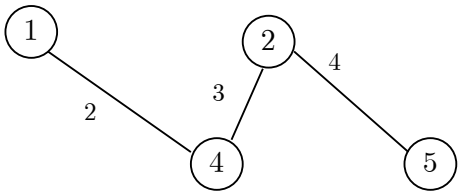
เส้นเชื่อมระหว่างโหนด 1 กับโหนด 2 มีค่าน้ำหนักเท่ากับ 7
เส้นเชื่อมระหว่างโหนด 4 กับโหนด 2 มีค่าน้ำหนักเท่ากับ 3
เส้นเชื่อมระหว่างโหนด 4 กับโหนด 3 มีค่าน้ำหนักเท่ากับ 10
เส้นเชื่อมระหว่างโหนด 4 กับโหนด 5 มีค่าน้ำหนักเท่ากับ 5



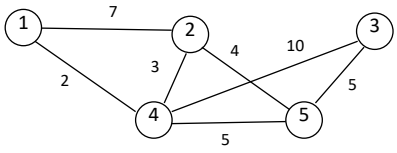
พิจารณาโหนดใน Spanning tree ซึ่งตอนนี้ใน Spanning tree มีโหนด 1 2 และ 4 ดังนั้นในการขยาย Spanning tree ให้พิจารณาโหนดที่เชื่อมต่อกับโหนด 1 2 และ 4 ซึ่งคือ โหนด 3 และ 5 (เฉพาะโหนดที่ไม่ได้อยู่ใน Spanning tree) แล้วเลือกโหนดที่เชื่อมต่อด้วยค่าน้ำหนักที่น้อยที่สุด



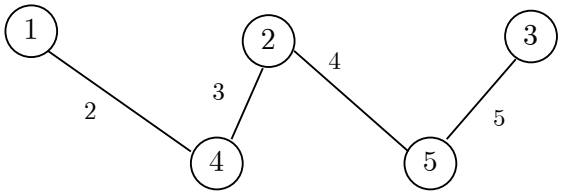
เส้นเชื่อมระหว่างโหนด 2 กับโหนด 5 มีค่าน้ำหนักเท่ากับ 4
เส้นเชื่อมระหว่างโหนด 4 กับโหนด 3 มีค่าน้ำหนักเท่ากับ 10
เส้นเชื่อมระหว่างโหนด 4 กับโหนด 5 มีค่าน้ำหนักเท่ากับ 5



พิจารณาโหนดใน Spanning tree ซึ่งตอนนี้ใน Spanning tree มีโหนด 1 2 4 และ 5 ดังนั้นในการขยาย Spanning tree ให้พิจารณาโหนดที่เชื่อมต่อกับโหนด 1 2 4 และ 5 ซึ่งมีโหนด 3 (เฉพาะโหนดที่ไม่ได้อยู่ใน Spanning tree) แล้วเลือกโหนดที่เชื่อมต่อด้วยค่าน้ำหนักที่น้อยที่สุด



เส้นเชื่อมระหว่างโหนด 4 กับโหนด 3 มีค่าน้ำหนักเท่ากับ 10
เส้นเชื่อมระหว่างโหนด 5 กับโหนด 3 มีค่าน้ำหนักเท่ากับ 5



สิ้นสุดการสร้าง Minimum spanning tree เนื่องจากจำนวนของโหนดใน Minimum spanning tree ครบแล้ว

คำถาม

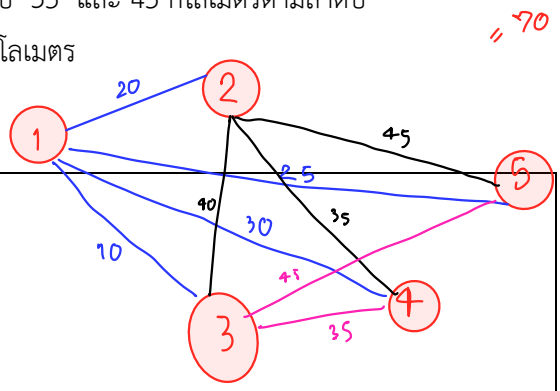
ประเทศแห่งหนึ่งมีลักษณะเป็นหมู่เกาะจำนวน 5 เกาะ ซึ่งแต่ละเกาะจะมีชื่อเรียกแตกต่างกันและมีระยะห่างระหว่างเกาะไม่เท่ากัน เมื่อมีระบบไฟฟ้าเข้าในประเทศท่านประธานาธิบดีต้องเชื่อมต่อไฟฟ้าให้ทุกเกาะสามารถใช้งานได้โดยไม่สนใจว่าจะต่อแบบไหนขอเพียงให้มีสายไฟฟ้าไปถึงยังทุกเกาะ แต่เนื่องด้วยท่านประธานาธิบดีต้องการประหยัดงบในการจัดซื้อสายไฟฟ้าจึงขอให้ท่านช่วยคิดว่าจะต้องซื้อสายไฟฟ้าน้อยที่สุดกี่กิโลเมตร

ระยะทางระหว่างเกาะ 1 กับเกาะ 2 3 4 และ 5 เท่ากับ 20 10 30 และ 25 กิโลเมตรตามลำดับ

ระยะทางระหว่างเกาะ 2 กับเกาะ 3 4 และ 5 เท่ากับ 40 35 และ 45 กิโลเมตรตามลำดับ

ระยะทางระหว่างเกาะ 3 กับเกาะ 4 และ 5 เท่ากับ 35 และ 45 กิโลเมตรตามลำดับ

ระยะทางระหว่างเกาะ 4 กับเกาะ 5 เท่ากับ 15 กิโลเมตร



Prim's Algorithm in C

```
1   #include<stdio.h>
2   #include<stdbool.h>
3
4   #define INF 9999999
5
6   // number of vertices in graph
7   #define V 5
8
9   // create a 2d array of size 5x5
10  //for adjacency matrix to represent graph
11  int G[V][V] = {
12    {0, 9, 75, 0, 0},
13    {9, 0, 95, 19, 42},
14    {75, 95, 0, 51, 66},
15    {0, 19, 51, 0, 31},
16    {0, 42, 66, 31, 0}};
17
18  int main() {
19    int no_edge;  // number of edge
20    int cost = 0;
21
22    // create an array to track selected vertex
23    // selected will become true otherwise false
24    int selected[V];
25
26    // set selected false initially
27    memset(selected, false, sizeof(selected));
28
29    // set number of edge to 0
30    no_edge = 0;
31
32    // the number of egde in minimum spanning tree will be
33    // always less than (V -1), where V is number of vertices in
34    //graph
35
36    // choose 0th vertex and make it true
37    selected[0] = true;
38
39    int x;  //  row number
40    int y;  //  col number
41
42    // print for edge and weight
43    printf("Edge : Weight\n");
```

```
44
45      while (no_edge < V - 1) {
46        //For every vertex in the set S, find the all adjacent vertices
47        // , calculate the distance from the vertex selected at step 1.
48        // if the vertex is already in the set S, discard it otherwise
49        //choose another vertex nearest to selected vertex  at step 1.
50                                  ค่าเยอะๆ
51        int min = INF;
52        x = 0;
53        y = 0;
54
55        for (int i = 0; i < V; i++) {
56          if (selected[i]) {
57            for (int j = 0; j < V; j++) {
58              if (!selected[j] && G[i][j]) {  // not in selected and there is an edge
59                if (min > G[i][j]) {
60                  min = G[i][j];
61                  x = i;
62                  y = j;
63                }
64              }
65            }
66          }
67        }
68        printf("%d - %d : %d\n", x, y, G[x][y]);
69        selected[y] = true;
70        no_edge++;
71        cost += G[x][y];
72      }
73      printf("\ncost : %d",cost);
74      return 0;
75    }
```

Prim's Algorithm in C++

```
1    /* Prim algorithm to find minimum spanning tree (in ubdirected graph)*/
2    #include <iostream>
3    #include <queue>
4    #include <vector>
5    #define MAXNODES 100
6    using namespace std;
7    struct edge {
8        int weight,node;
9        bool operator <(const edge &a) const
10       {
11           return weight>a.weight;
12       }
13   };
14   vector <edge> adj[MAXNODES];
15   priority_queue <edge> Q;
16   int nodesN,edgesN;
17   int prim(int x);
18
19   int main() {
20       int x,y,weight;
21       int minimumcost;
22       edge a;
23
24       cin >> nodesN >> edgesN;
```

```
25        for (int i=0;i<edgesN;i++) {
26            cin >> x >> y >> weight;
27            a.weight = weight;
28            a.node = x;
29            adj[y].push_back(a);
30            a.node = y;
31            adj[x].push_back(a);
32        }
33        minimumcost = prim(1);
34        cout << minimumcost;
35        return 0;
36    }
37
38    int prim(int x) {
39        edge a;
40        int idx;
41        int cost = 0;
42        int countNodes = 1;
43        bool marked[MAXNODES] = {0};
44        while (!adj[x].empty()) {
45            a = adj[x].back();
46            adj[x].pop_back();
47            Q.push(a);
48        }
49        marked[x] = true;
50        while (!Q.empty()) {
51                a = Q.top();
52                Q.pop();
53                if (marked[a.node] == true) continue;
54                cost += a.weight;
55                marked[a.node] = true;
56                countNodes++;
57
58                if (countNodes== nodesN) return cost;
59                idx = a.node;
60                while (!adj[idx].empty()) {
61                    a = adj[idx].back();
62                    adj[idx].pop_back();
63                    Q.push(a);
64                }
65
66        }
67        return cost;
68
69    }
70
71
```

สร้าง กราฟ

→ถ้าถูกนำไปสร้างแล้ว จะขึ้นไป Q อีกไม่ได้

นำเส้นเชื่อมต่อทั้งหมดไปเก็บใน Q

Kruskal's algorithm

แนวคิดของ Kruskal's algorithm คือ เริ่มจากการเรียงลำดับเส้นเชื่อมตามค่าน้ำหนักจากน้ำหนักน้อยที่สุดไปยังมากที่สุด หลังจากนั้นก็จะทำการพิจารณาเส้นเชื่อมตามลำดับด้วยการนำโหนดทั้งสองของเส้นเชื่อมมาสร้าง Spanning tree โดยโหนดทั้งสองจะต้องไม่ทำให้เกิด Cycle ใน Spanning tree ถ้าโหนดของเส้นเชื่อมใดทำให้เกิด Cycle ก็จะไม่นำเส้นเชื่อมนั้นมาสร้าง Spanning tree หลังจากนั้นก็ทำการพิจารณาเส้นเชื่อมถัดไปเรื่อยๆ จนกระทั่งได้จำนวนเส้นเชื่อมของ Spanning tree เท่ากับจำนวนโหนดของกราฟลบหนึ่ง

The steps for implementing Kruskal's algorithm are as follows:

1. Sort all the edges in increasing order of their weight.
2. Pick the smallest weight edge and check it to form a cycle with the spanning-tree formed. If the cycle is not formed, include this edge to the spanning tree. Else, discard it.
3. Repeat step 2 until we get (V-1) edges in the spanning tree.

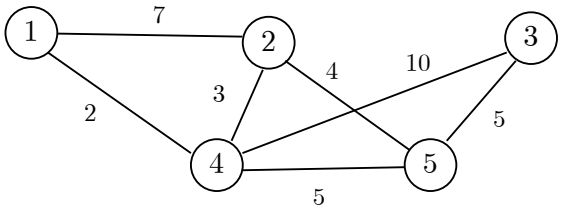
Union–Find Algorithm for cycle detection in a graph

1. Create disjoint sets for each vertex of the graph.
2. For every edge u, v in the graph

    i) Find the root of the sets to which elements u and v belongs.

    ii) If both u and v have the same root in disjoint sets, a cycle is found.

ตัวอย่าง การใช้ Union–Find Algorithm for cycle detection in a graph



โหนดรากของโหนด 1 คือ โหนด 1
โหนดรากของโหนด 2 คือ โหนด 2
โหนดรากของโหนด 3 คือ โหนด 3
โหนดรากของโหนด 4 คือ โหนด 4
โหนดรากของโหนด 5 คือ โหนด 5

เลือกเส้นเชื่อมระหว่างโหนด 1 กับ 2

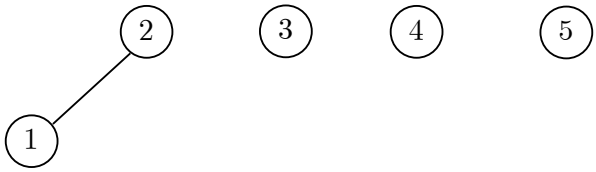
จากการตรวจสอบโหนดราก

จะได้ว่า

โหนดรากของโหนด 1 คือ โหนด 1
โหนดรากของโหนด 2 คือ โหนด 2

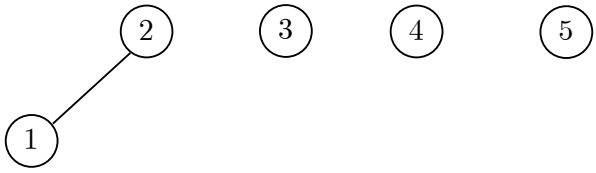
ไม่เกิด **Cycle graph**

ดังนั้นจะได้ว่า โหนดรากของโหนด 1 คือ โหนด 2



เลือกเส้นเชื่อมระหว่างโหนด 2 กับ 5

จากการตรวจสอบโหนดราก

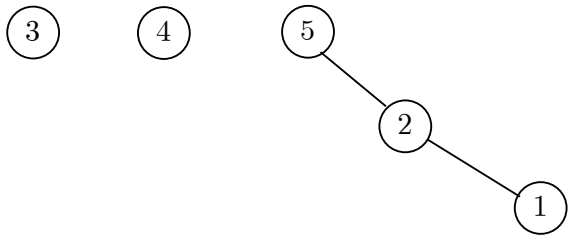


จะได้ว่า

โหนดรากของโหนด 2 คือ โหนด 2
โหนดรากของโหนด 5 คือ โหนด 5

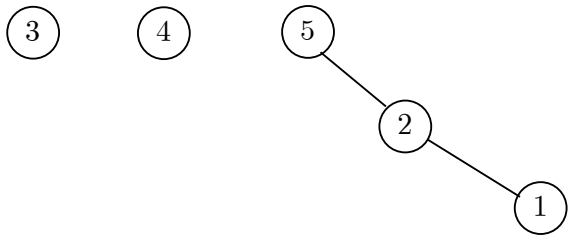
ไม่เกิด **Cycle graph**

ดังนั้นจะได้ว่า โหนดรากของโหนด 2 คือ โหนด 5



เลือกเส้นเชื่อมระหว่างโหนด 2 และ 4

จากการตรวจสอบโหนดราก



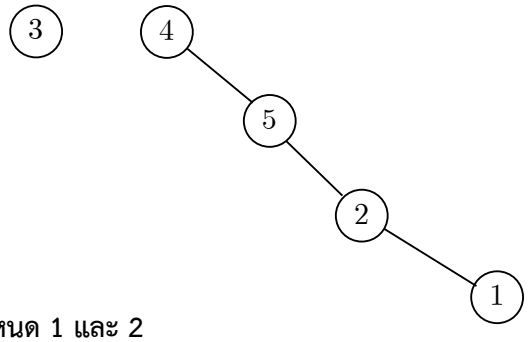
จะได้ว่า

โหนดรากของโหนด 2 คือ โหนด 5

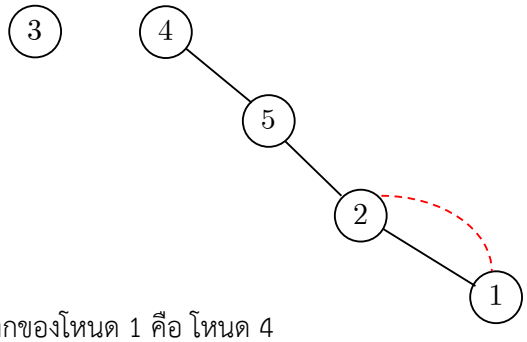
โหนดรากของโหนด 4 คือ โหนด 4

ไม่เกิด **Cycle graph**

ดังนั้นจะได้ว่า โหนดรากของโหนด 5 คือ โหนด 4



เลือกเส้นเชื่อมระหว่างโหนด 1 และ 2



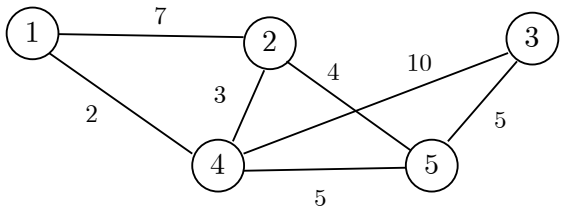
โหนดรากของโหนด 1 คือ โหนด 4

โหนดรากของโหนด 2 คือ โหนด 4

เกิด **Cycle graph**

```
int getRoot(int v,int root[]) {
    if (root[v]==v) {
        return v;
    }
    return getRoot(root[v],root);
}


//Union find to detect cycle
for (int i=0;i<=nodesN;i++) {
      parent[i] = i;
}


p1 = getRoot(currentEdge.src,root);
p2 = getRoot(currentEdge.dest,root);

if (p1!=p2)
   // not detect cycle graph
   parent[p1] = p2;
   ……

else
   // Detect cycle graph
   …..
```

ตัวอย่าง การสร้าง Minimum spanning tree ของกราฟ G ด้วย Kruskal's algorithm



เรียงลำดับเส้นเชื่อมตามค่าน้ำหนักจากน้อยไปมาก

เส้นเชื่อมระหว่างโหนด 1 กับโหนด 4 มีค่าน้ำหนักเท่ากับ 2
เส้นเชื่อมระหว่างโหนด 2 กับโหนด 4 มีค่าน้ำหนักเท่ากับ 3
เส้นเชื่อมระหว่างโหนด 2 กับโหนด 5 มีค่าน้ำหนักเท่ากับ 4
เส้นเชื่อมระหว่างโหนด 3 กับโหนด 5 มีค่าน้ำหนักเท่ากับ 5
เส้นเชื่อมระหว่างโหนด 4 กับโหนด 5 มีค่าน้ำหนักเท่ากับ 5
เส้นเชื่อมระหว่างโหนด 1 กับโหนด 2 มีค่าน้ำหนักเท่ากับ 7
เส้นเชื่อมระหว่างโหนด 3 กับโหนด 4 มีค่าน้ำหนักเท่ากับ 10

สร้างโหนดรากของแต่ละโหนด จะได้ว่า

โหนดรากของโหนด 1 คือ โหนด 1 โหนดรากของโหนด 2 คือ โหนด 2
โหนดรากของโหนด 3 คือ โหนด 3 โหนดรากของโหนด 4 คือ โหนด 4
โหนดรากของโหนด 5 คือ โหนด 5

พิจารณาเส้นเชื่อม

เส้นเชื่อมระหว่างโหนด 1 กับโหนด 4 มีค่าน้ำหนักเท่ากับ 2 ✓
เส้นเชื่อมระหว่างโหนด 2 กับโหนด 4 มีค่าน้ำหนักเท่ากับ 3
เส้นเชื่อมระหว่างโหนด 2 กับโหนด 5 มีค่าน้ำหนักเท่ากับ 4
เส้นเชื่อมระหว่างโหนด 3 กับโหนด 5 มีค่าน้ำหนักเท่ากับ 5
เส้นเชื่อมระหว่างโหนด 4 กับโหนด 5 มีค่าน้ำหนักเท่ากับ 5
เส้นเชื่อมระหว่างโหนด 1 กับโหนด 2 มีค่าน้ำหนักเท่ากับ 7
เส้นเชื่อมระหว่างโหนด 3 กับโหนด 4 มีค่าน้ำหนักเท่ากับ 10

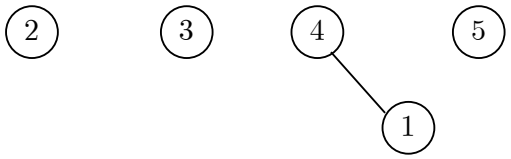
เลือกเส้นเชื่อมระหว่างโหนด 1 กับโหนด 4 แล้วตรวจสอบโหนดรากของโหนดทั้งสอง
จากการตรวจสอบโหนดราก

จะได้ว่า

โหนดรากของโหนด 1 คือ โหนด 1
โหนดรากของโหนด 4 คือ โหนด 4

ไม่เกิด **Cycle graph**

ดังนั้นจะได้ว่า โหนดรากของโหนด 1 คือ โหนด 4

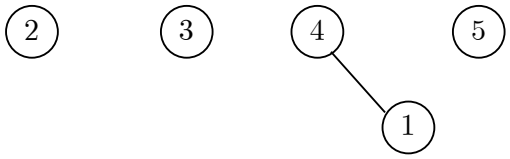


ตรวจสอบ cycle

พิจารณาเส้นเชื่อมถัดไป

เส้นเชื่อมระหว่างโหนด 1 กับโหนด 4 มีค่าน้ำหนักเท่ากับ 2 ✓
เส้นเชื่อมระหว่างโหนด 2 กับโหนด 4 มีค่าน้ำหนักเท่ากับ 3 ✓
เส้นเชื่อมระหว่างโหนด 2 กับโหนด 5 มีค่าน้ำหนักเท่ากับ 4
เส้นเชื่อมระหว่างโหนด 3 กับโหนด 5 มีค่าน้ำหนักเท่ากับ 5
เส้นเชื่อมระหว่างโหนด 4 กับโหนด 5 มีค่าน้ำหนักเท่ากับ 5
เส้นเชื่อมระหว่างโหนด 1 กับโหนด 2 มีค่าน้ำหนักเท่ากับ 7
เส้นเชื่อมระหว่างโหนด 3 กับโหนด 4 มีค่าน้ำหนักเท่ากับ 10

เลือกเส้นเชื่อมระหว่างโหนด 2 กับโหนด 4 และจากการตรวจสอบโหนดราก

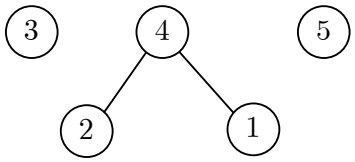


จะได้ว่า

โหนดรากของโหนด 2 คือ โหนด 2
โหนดรากของโหนด 4 คือ โหนด 4

ไม่เกิด **Cycle graph**

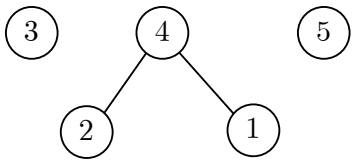
ดังนั้นจะได้ว่า โหนดรากของโหนด 2 คือ โหนด 4



พิจารณาเส้นเชื่อมถัดไป

เส้นเชื่อมระหว่างโหนด 1 กับโหนด 4 มีค่าน้ำหนักเท่ากับ 2 ✓
เส้นเชื่อมระหว่างโหนด 2 กับโหนด 4 มีค่าน้ำหนักเท่ากับ 3 ✓
เส้นเชื่อมระหว่างโหนด 2 กับโหนด 5 มีค่าน้ำหนักเท่ากับ 4 ✓
เส้นเชื่อมระหว่างโหนด 3 กับโหนด 5 มีค่าน้ำหนักเท่ากับ 5
เส้นเชื่อมระหว่างโหนด 4 กับโหนด 5 มีค่าน้ำหนักเท่ากับ 5
เส้นเชื่อมระหว่างโหนด 1 กับโหนด 2 มีค่าน้ำหนักเท่ากับ 7
เส้นเชื่อมระหว่างโหนด 3 กับโหนด 4 มีค่าน้ำหนักเท่ากับ 10

เส้นเชื่อมระหว่างโหนด 2 กับโหนด 5 และจากการตรวจสอบโหนดราก

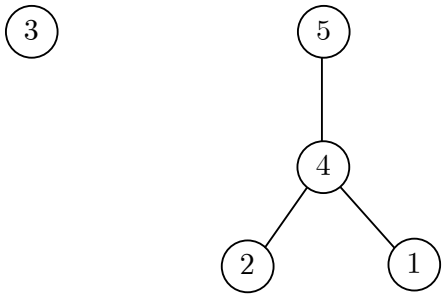


จะได้ว่า

โหนดรากของโหนด 2 คือ โหนด 4
โหนดรากของโหนด 5 คือ โหนด 5

ไม่เกิด **Cycle graph**

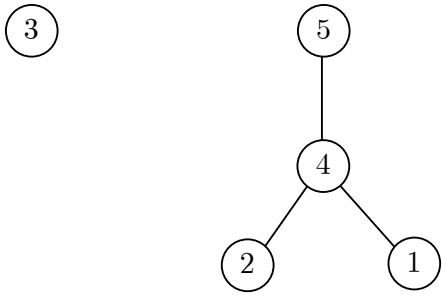
ดังนั้นจะได้ว่า โหนดรากของโหนด 4 คือ โหนด 5



พิจารณาเส้นเชื่อมถัดไป

เส้นเชื่อมระหว่างโหนด 1 กับโหนด 4 มีค่าน้ำหนักเท่ากับ 2 ✓
เส้นเชื่อมระหว่างโหนด 2 กับโหนด 4 มีค่าน้ำหนักเท่ากับ 3 ✓
เส้นเชื่อมระหว่างโหนด 2 กับโหนด 5 มีค่าน้ำหนักเท่ากับ 4 ✓
เส้นเชื่อมระหว่างโหนด 3 กับโหนด 5 มีค่าน้ำหนักเท่ากับ 5 ✓
เส้นเชื่อมระหว่างโหนด 4 กับโหนด 5 มีค่าน้ำหนักเท่ากับ 5
เส้นเชื่อมระหว่างโหนด 1 กับโหนด 2 มีค่าน้ำหนักเท่ากับ 7
เส้นเชื่อมระหว่างโหนด 3 กับโหนด 4 มีค่าน้ำหนักเท่ากับ 10

เส้นเชื่อมระหว่างโหนด 3 กับโหนด 5 และจากการตรวจสอบโหนดราก

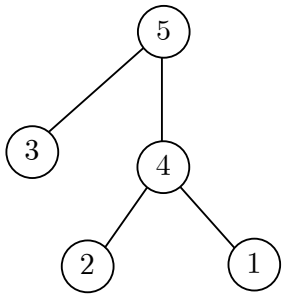


จะได้ว่า

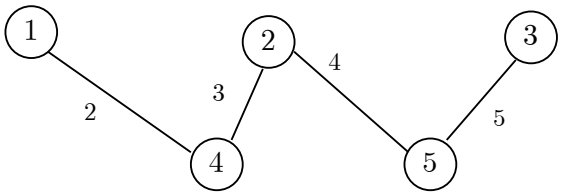
โหนดรากของโหนด 3 คือ โหนด 3
โหนดรากของโหนด 5 คือ โหนด 5

ไม่เกิด **Cycle graph**

ดังนั้นจะได้ว่า โหนดรากของโหนด 3 คือ โหนด 5



จบการสร้าง Minimum spanning tree เนื่องจากจำนวนเส้นเชื่อมเท่ากับจำนวนของกราฟ G ลบหนึ่ง ดังนั้นจะได้ Minimum spanning tree ดังต่อไปนี้



คำถาม

ประเทศแห่งหนึ่งมีลักษณะเป็นหมู่เกาะจำนวน 5 เกาะ ซึ่งแต่ละเกาะจะมีชื่อเรียกแตกต่างกันและมีระยะห่างระหว่างเกาะไม่เท่ากัน เมื่อมีระบบไฟฟ้าเข้าในประเทศท่านประธานาธิบดีต้องเชื่อมต่อไฟฟ้าให้ทุกเกาะสามารถใช้งานได้โดยไม่สนใจว่าจะต่อแบบไหนขอเพียงให้มีสายไฟฟ้าไปถึงยังทุกเกาะ แต่อย่างไรก็ตามสำหรับระหว่างเกาะบางคู่จะไม่สามารถลากสายไฟฟ้าได้ระหว่างกันได้เพราะติดแนวปะการัง ดังนั้นเนื่องจากท่านประธานาธิบดีต้องการประหยัดงบในการจัดซื้อสายไฟฟ้าจึงขอให้ท่านช่วยคิดว่าจะต้องซื้อสายไฟฟ้าน้อยที่สุดกี่กิโลเมตร โดยระยะทางที่สามารถลากสายไฟฟ้าได้เป็นดังนี้

ระยะทางระหว่างเกาะ 1 กับเกาะ 2 4 และ 5 เท่ากับ 10 30 และ 25 กิโลเมตรตามลำดับ

ระยะทางระหว่างเกาะ 2 กับเกาะ 3 และ 5 เท่ากับ 35 และ 15 กิโลเมตรตามลำดับ

ระยะทางระหว่างเกาะ 3 กับเกาะ 4 และ 5 เท่ากับ 40 และ 45 กิโลเมตรตามลำดับ

ระยะทางระหว่างเกาะ 4 กับเกาะ 5 เท่ากับ 25 กิโลเมตร

```
1   /* Kruskal algorithm to find minimum spanning tree (in ubdirected graph)
2   Cycle detection using union find algorithm */
3   #include <iostream>
4   #include <queue>
5   #include <vector>
6   #define MAXNODES 100
7   using namespace std;
8   struct edge {
9       int weight,src,dest;
10      bool operator <(const edge &a) const
11      {
12          return weight>a.weight;
13      }
14  };
15
16  priority_queue <edge> Q;
17  int nodesN,edgesN;
18
19  int getParent(int v,int parent[]) {
20      if (parent[v]==v) {
21          return v;
22      }
23      return getParent(parent[v],parent);
24  }
25
26  int main() {
27      int x,y,weight;
28      int minimumcost = 0, countNode = 0;
29      edge a;
30      cin >> nodesN >> edgesN;
31      int parent[nodesN];
32      edge currentEdge;
33
34      for (int i=0;i<edgesN;i++) {
35          cin >> x >> y >> weight;
36          a.weight = weight;
37          a.src = x;
38          a.dest = y;
39          Q.push(a);
40      }
41      //Union find to detect cycle
42      for (int i=0;i<=nodesN;i++) {
43          parent[i] = i;
44      }
45
46      while (countNode<nodesN-1 && !Q.empty()) {
47          currentEdge = Q.top();
48          Q.pop();
49          int p1 = getParent(currentEdge.src,parent);
50          int p2 = getParent(currentEdge.dest,parent);
51          if (p1!=p2) {
52              countNode++;
53              parent[p1] = p2;
54              minimumcost += currentEdge.weight;
55          //    cout << currentEdge.src << "  " << currentEdge.dest<<endl;
56          }
57      }
58      cout << minimumcost;
59      return 0;
60  }
61
```

สร้างกราฟ

หา cycle

กำหนด root node ของแต่ละโหนด